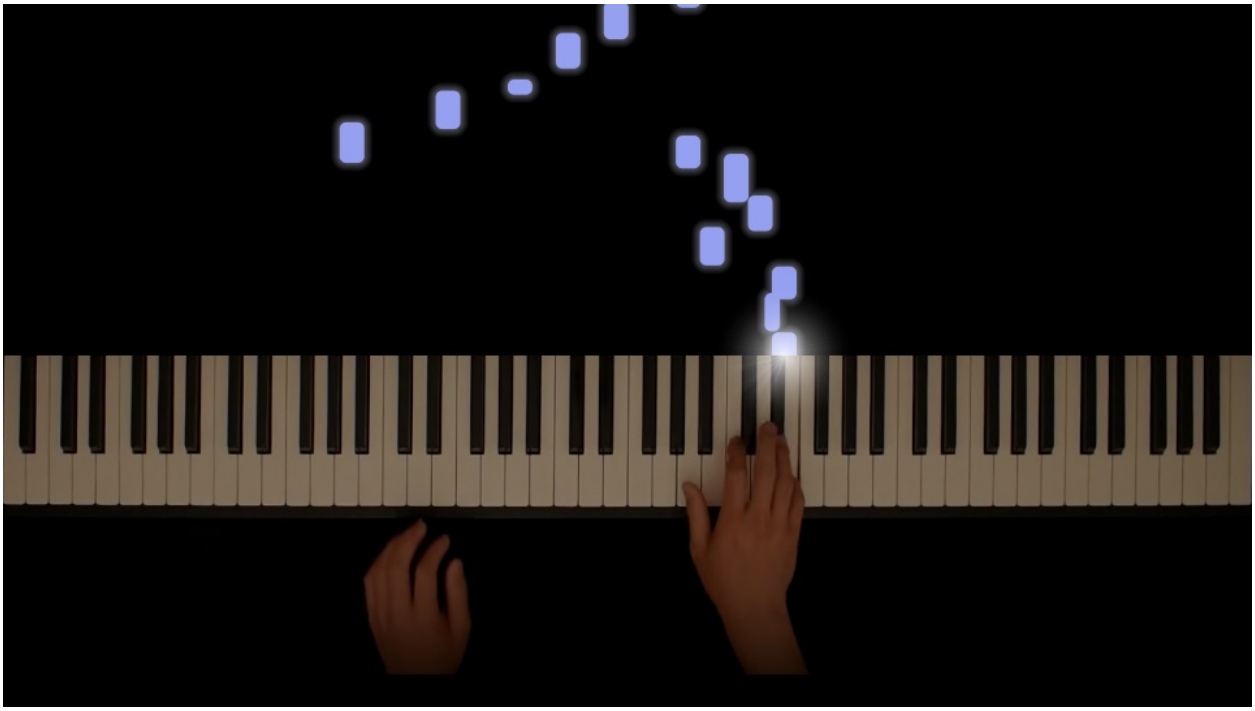
PianoRay

Release 0.1.1

PianoRay Authors

May 03, 2022

GENERAL

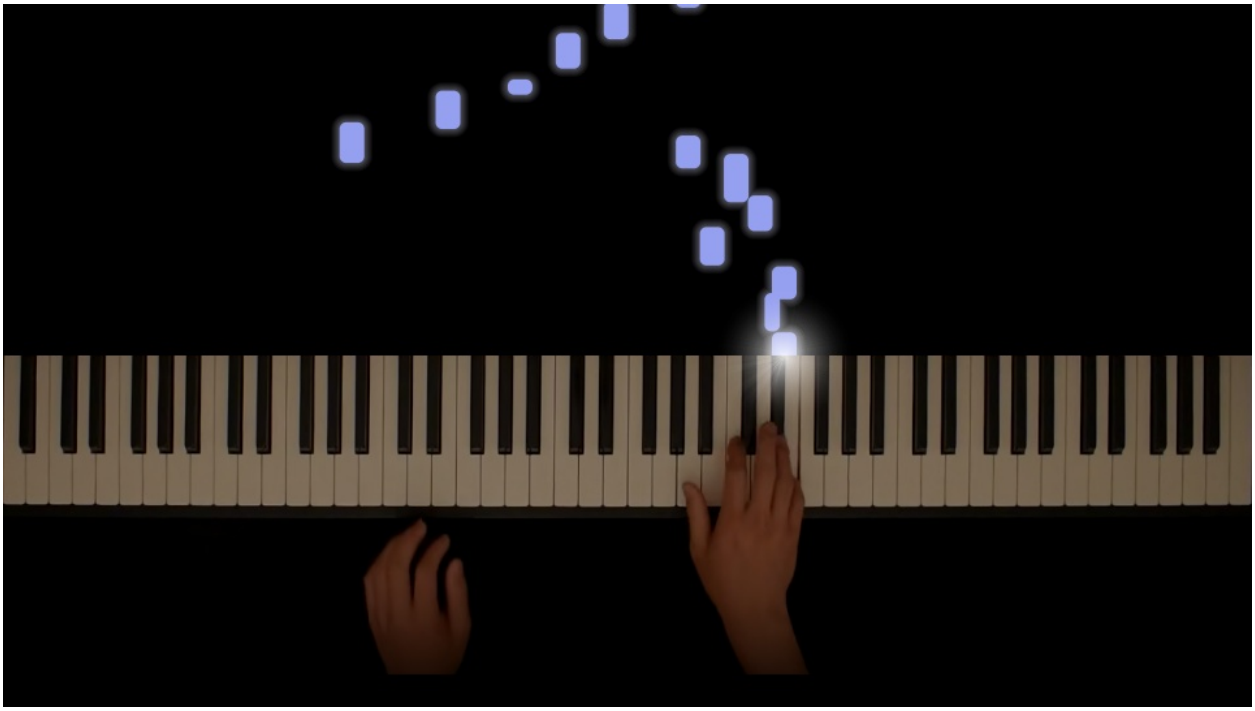


CHAPTER

ONE

ABOUT

PianoRay is a piano visualization tool.



1.1 Features

- Adds dropping blocks.
- Crops piano from video.
- Glare when blocks hit piano.
- Automatically compiles audio and video.

CHAPTER

TWO

INSTALLATION

2.1 Dependencies

- Python version 3.8 or higher.
- FFmpeg.
- C++ compiler (`g++`).
- Python packages listed in `requirements.txt`

2.2 Latest

PianoRay is distributed on PyPI. Install with

```
pip install pianoray
```

2.3 Master Branch

May be unstable or have bugs.

```
pip install git+https://github.com/phuang1024/pianoray
```

CHAPTER

THREE

SUPPORT

Please open an issue or discussion on GitHub.

CHAPTER

FOUR

LICENSE

PianoRay is licensed under GNU GPL v3. See `LICENSE` for the full license text.

You are free to use the software however you want.

If you are using the code itself, i.e. using some code from PianoRay for your own projects, you must license the complete derived work with a compatible license.

If you are just using the output i.e. the rendered videos, you may use the output however you want, with or without credit.

CHAPTER

FIVE

FIRST VIDEO

First, install PianoRay. Follow instructions in General/Installation.

5.1 Example Files

Download example performance files. This script copies the Fur Elise example recording to `~/pianoray_tutorial`.

```
cd /tmp
git clone https://github.com/phuang1024/pianoray
cd pianoray/examples/furelise

mkdir ~/pianoray_tutorial
cp video.mp4 midi.mid audio.mp3 ~/pianoray_tutorial
```

The video file contains the recording of the piano. The MIDI file contains data about which notes are played. The audio file has the audio.

5.2 Create Settings

PianoRay reads settings from a JSON file. This file contains nested mappings with `key: value` pairs.

Save this data to `~/pianoray_tutorial/settings.json`:

```
{
    "video": {
        "resolution": [1280, 720],
        "fps": 30
    },
    "midi": {
        "file": "midi.mid"
    },
    "audio": {
        "file": "audio.mp3",
        "start": 20.74
    },
    "keyboard": {
        "file": "video.mp4",
        "start": 4.75,
        "crop": [[252,480], [1793,487], [1789,676], [257,666]],
```

(continues on next page)

(continued from previous page)

```
        "dim_mult": 0.6,
        "dim_add": -8
    }
}
```

Most of these settings are self explanatory.

The `start` settings are timestamps, in seconds, of when you press the first note in the respective media. This is necessary to apply the correct offsets.

The `crop` setting defines the coordinates of the keyboard in the video. These values are for the Fur Elise recording.

5.3 Render

To start the render, run these commands in a shell:

```
cd ~/pianoray_tutorial
pianoray render -s settings.json -o out.mp4 -p
```

This starts rendering, using the provided settings file and saving to the output file. The `-p` flag tells PianoRay to open the output file after rendering.

Rendering may take a few minutes. If the renderer crashes, run the same command again. If it repeatedly does not work, open an issue on GitHub.

CHAPTER

SIX

RECORDING

Instructions for recording and making your own video.

6.1 Recording

You will need to record two files: Video and MIDI. In order to record these files, you will need a MIDI keyboard, a camera, and a computer.

6.1.1 Video

Find a setup with a camera looking down vertically onto the keyboard. Some things to consider:

- Safety: Make sure the camera won't fall down.
- Stability: Try to reduce shaking, e.g. from vibrations from the keyboard.
- Focus: Make sure you focus the camera onto the keyboard before recording. It is very disappointing to find that the video is ruined because the keyboard recording is blurry (speaking from experience).
- Background: If you desire, place a dark tarp under the keyboard so you can create the "hands floating over nothing" effect. There are some settings in PianoRay to dim the background and achieve this effect.
- Privacy: If you plan to release the video to the public, make sure it doesn't contain any private information.

6.1.2 MIDI

Connect the MIDI keyboard to the computer. Use MIDI recording software to record the MIDI. I use MidiEditor, which has worked great.

6.2 Processing

6.2.1 Audio

Create an audio file from the MIDI.

1. Download a soundfont. SoundFonts4U has great piano soundfonts.
2. Install software that can render a MIDI file. I use FluidSynth, and the rest of these instructions assume you have FluidSynth.

3. Run this command, which uses FluidSynth to render and FFmpeg to write the audio file: `fluidsynth -a alsa -T raw -g GAIN -F - SOUNDFONT.sf2 MIDI.mid | ffmpeg -y -f s32le -i - -filter:a "volume=2" AUDIO.mp3`. Replace the uppercase words with the respective values. A value of `0.5` for GAIN works usually.

6.2.2 Video

Make sure the video is right side up. That is, your hands come from the bottom of the screen and play the keyboard.

If you need to rotate it, see this page for rotating with FFmpeg.

6.3 Offsets

Find the offsets for respective media. PianoRay uses these offsets. It may be beneficial to write down these offsets somewhere so you don't forget them later.

6.3.1 Audio

Open the audio in an audio player and find the timestamp, in seconds, when the audio starts. I use Audacity.

6.3.2 Video

Find the moment you play the first note in the video. I use Blender's video editor.

6.3.3 Video Crop

Find the pixel coordinates of the four corners of the keyboard in the video, starting from the top left and going clockwise. If you use Blender's video editor, keep in mind that Blender's image viewer has the Y coordinates reversed.

6.4 Rendering

Follow instructions in this page for rendering instructions.

CHAPTER

SEVEN

CLI

Command line interface arguments.

Type `pianoray -h` for info.

7.1 Example Commands

- Render: `pianoray render -s settings.json -o out.mp4`
- Render and play: `pianoray render -s settings.json -o out.mp4 -p`

7.2 Resume Previous Render

While rendering, PianoRay saves which frame is currently being rendered to the cache. This allows resuming a render if it is interrupted.

Configure render resuming with the `--resume=...` flag.

- If omitted, PianoRay will ask via stdin if you wish to resume.
- If `True`, PianoRay will always resume if a previous render exists.
- If `False`, PianoRay will never resume.

If the previous render finished completely, you can pass `--resume=True` to only recompile the frames into a video.

CHAPTER

EIGHT

SETTINGS

When creating a video, many settings are passed to PianoRay. The settings should be stored in a JSON file containing dictionaries of `name: value` pairs with nested dictionaries.

8.1 Example

Save a JSON file of your settings, and pass to `pianoray`.

Example:

```
{
    "video": {
        "fps": 30,
        "resolution": [1920, 1080]
    },
    "midi": {
        "path": "/path/to/file.mid"
    }
}
```

8.2 Available

Most settings have default values. See `src/utils.py` for the default settings.

See devs/info to understand what a "coord" is.

Colors are `[R, G, B]` from 0 to 255.

- **`video`: Settings about video export.**
  - `fps`: Frames per second.
  - `resolution`: Video resolution `[width, height]`.
  - `vcodec`: Video codec. This will be passed to FFmpeg, so please provide a value that FFmpeg recognizes, e.g. `libx264`.
- **`audio`: Settings about audio.**
  - `file`: Path to audio file.
  - `start`: Timestamp, in seconds, you play the first note in the audio.
- **`composition`: Settings about the structure of the video.**

  - `margin_start`: Seconds of video before the first note.
  - `margin_end`: Seconds of video after the last note ends.
  - `fade_in`: Seconds of fade in.
  - `fade_out`: Seconds of fade out.
  - `fade_blur`: Fade blur radius in coords.
- **`piano`: Settings about the piano.**
  - `black_width_fac`: Black key width as factor of white key width.
- **`blocks`: Settings about blocks.**
  - `speed`: If `X` is the distance between the top of the screen and the top of the keyboard, the blocks travel `speed * X` per second.
  - `color`: RGB color of the blocks.
  - `radius`: Block corner rounding radius in coords.
  - `glow_intensity`: Intensity of glow around the block.
  - `glow_color`: Color of glow.
  - `glow_radius`: Radius of glow in coords.
- **`midi`: Settings about MIDI.**
  - `file`: Path to MIDI file.
  - `speed`: Speed multiplier.
  - `min_length`: Minimum note length in seconds.
- **`keyboard`: Settings about rendering the keyboard.**
  - `file`: Video file containing recording of playing the keyboard.
  - `start`: Timestamp, in seconds, you play the first note in the video.
  - `crop`: Corners of the keyboard in the video, starting from top left and going clockwise. Should be `[[x1, y1], [x2, y2], [x3, y3], [x4, y4]]`.
  - `dim_mult`: Multiplicative dimming. Give a multiplier value.
  - `dim_add`: Additive dimming. Subtracted from 0 to 255 channels. If you wish to dim, provide a negative value.
  - `below_length`: Length of the below section in coords.
  - `octave_lines`: Whether to render octave lines.
- **`glare`: Settings about rendering glare.**
  - `radius`: Radius of glare in coords.
  - `intensity`: Intensity value.
  - `jitter`: Random intensity change for each frame.
  - `streaks`: Number of streaks per glare.

CHAPTER

NINE

SETUP

How to setup your development environment.

9.1 Dependencies

See dependencies in the `General/Installation` page.

Additional dependencies for development:

- Git
- GNU Make

9.2 Fork and Clone

First, fork the GitHub repository and clone your fork.

9.3 Test Video

```
cd /path/to/pianoray
make wheel
make install
pianoray render -s tests/furelise.json -o out.mp4 -p
```

This should render the video and open it in your video player. Rendering may take a few minutes.

CHAPTER

TEN

FILE STRUCTURE

Information about the project files.

10.1 /

Root directory. Contains cool files.

Python module `setup.py` and `MANIFEST.in` are here.

There is a Makefile with convenient targets.

10.2 .github

GitHub files, like workflows.

10.3 docs

Documentation. Docs are generated with Python sphinx and hosted on ReadTheDocs.

10.4 examples

Example recordings for testing.

10.5 pianoray

Source code for everything.

10.5.1 pianoray/

Main module and global utilities.

- `__init__.py`: Module file.
- `cpp.py`: Handles C++ library compiling and loading.
- `logger.py`: Logging utilities.
- `main.py`: Main entry point.
- `settings.py`: Class for convenient settings access.
- `utils.py`: Global utilities.

10.5.2 pianoray/cutils

C++ header files for C++ libraries.

- `pr_image.hpp`: Image class for interacting with raw unsigned char data.
- `pr_math.hpp`: Math utilities.
- `pr_piano.hpp`: Utilities relating to rendering, e.g. piano dimensions.
- `pr_random.hpp`: Random number generator utilities.

10.5.3 pianoray/effects

Files that render the video.

- `effect.py`: Effect base class for OOP internally.
- `midi.py`: Parse MIDI.
- `blocks.py`, `blocks.cpp`: Rendering blocks.
- `glare.py`, `glare.cpp`: Rendering glare.
- `keyboard.py`: Rendering keyboard.

10.5.4 pianoray/render

Rendering pipeline.

- `render.py`: Calling effects to render the video.
- `video.py`: Class for managing video frames and calling FFmpeg to compile the video.
- `lib.py`: Load and initialize C++ libraries.

10.5.5 pianoray/view

PianoRay viewer files. Currently in development.

10.6 scripts

Small scripts, like style checks.

10.7 tests

Testing files, like test JSON settings.

CHAPTER

ELEVEN

CACHE

PianoRay stores temporary files in a cache directory (default `.prcache`). The cache can be safely deleted at any time.

11.1 File Structure

- `./c_libs`: Compiled C library object and library files.
- `./output`: Output render is stored here.
- `./settings.json`, `./currently_rendering.txt`: Files that store the state of the rendering. This is used to resume rendering if desired. See CLI for more info.

CHAPTER

TWELVE

INFORMATION

Just information for now.

- Frame zero is when the first note begins.
- Note zero is lowest note on piano.
- One `coord` (unit of distance) is the width of one white key in the video. This is equal to the horizontal resolution divided by 52. For example, for a 1920x1080 video, one coord is `36.924` pixels.